A Comparative Analysis of Multiplexer Techniques For the Minimization of Function Cost Using the Costtable Approach*

Jon T. Butler
Dept. of Electr. and Comp. Eng.
Naval Postgraduate School, Code EC/Bu
Monterey, CA 93943-5004 U.S.A.

Hans G. Kerkhoff
IC Technology and Electronics Gr.
University of Twente
7500 AE Enschede, The Netherlands

Siep Onneweer
Nederlandse Philips Bedrijven, CFT/CADE
HKJ 1810 Kastanjestraat, Postbox 218
5600 MD Eindhoven, The Netherlands

ABSTRACT

In the costtable approach to logic design, a given function is realized by selecting functions from a table and combining them. Associated with each function is a cost, and the goal is to find, among all realizations, the one of least cost. We show an extension to the costtable approach in which functions are combined using a multiplexer, with the goal to find an arrangement of the functions that yields the lowest cost. Specifically, we analyze two techniques to minimize total function cost 1) choosing which variables to apply to the multiplexer inputs and 2) choosing a permutation of logic values that yields lowest cost. We analyze the relative benefits of 1), 2), and 1) and 2) together. Our basis of comparison is a set of randomly chosen two- and three-variable 4-valued functions. We show that these techniques yield a reduction of 7 to 34% in the average cost over the use of a multiplexer without such techniques.

I. INTRODUCTION

Because of compactness of logic circuits, there is considerable interest in multiple-valued CCD (charge-coupled devices) technology [3, 4]. Of the possible design approaches for MVL-CCD, the costtable approach has received the most attention [1-2, 5-6, 8-10]. In addition, the costtable approach has been used in the design of current-mode CMOS circuits [7] In the costtable approach, a given function is realized by selecting functions from a table and combining them. Each chosen function has a cost, as does the combining operation. Cost can represent chip area, power dissipation, speed, etc.. The goal is to find low cost functions which, when combined, yield the given function, and to use as few functions as possible.

Kerkhoff and Robroek [5] introduce the costtable technique for the synthesis of one-variable 4-valued functions implemented in CCD. The table contains 45 functions, from which all 256 one-variable functions are synthesized. The cost of each function is an approximation to the chip area occupied by a CCD realization of that function. Lee and Butler [6] show a costtable of 24 entries that produces realizations as good as or better than those in [5]. The choice of a costtable is determined by the total cost of the realizations produced; for a given costtable size, one wants a costtable that yields the lowest total cost. Schueller, Tirumalai, and Butler [10] show minimal and near-minimal costtables, and from this, find that the costtable of [6] is not minimal. Also, it is observed that there is a point of diminishing returns with respect to costtable size. That is, while costtables of larger size produce more economical realizations, beyond a certain size, about 10% of the total number of functions to be synthesized, there is little benefit to adding more functions to the costtable.

Schueller and Butler [9] show that the *average* costtable is significantly less efficient than the optimal one for small costtables, but close to the optimal one for large costtables. In addition, it is shown that a search for minimal costtables cannot exclude certain seemingly useless functions, called composite functions that are more efficiently realized by summing other functions. Schueller and Butler [10] show that design by costtable is an NP-complete problem. Abd-El-Barr, Hoang, and Vranesic [2] show further improvements by exhaustively searching through realizations for the one of least cost.

In this paper, we consider the use of multiplexers in the realization of functions, where the primary inputs are costtable functions. Specifically, we analyze the relative benefits of 1) choosing which function variable to apply to the primary inputs, of 2) permuting variable logic values, and of both 1) and 2). Our analysis is over a set of 5000 randomly generated two- and three-variable 4-valued

*Research supported by NATO Grant 423/84, NSF Grant MIP-8706553, the Naval Research Laboratory under a block funded grant through the Naval Postgraduate School, and in part by the Dutch Innovative Research Program (IOP).

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 1990		

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Comparative Analysis of Multiplexer Techniques For the Minimization of Function Cost Using the Costtable Approach	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In the costtable approach to logic design, a given function is realized by selecting functions from a table and combining them. Associated with each function is a cost and the goal is to find, among all realizations, the one of least cost. We show an extension to the costtable approach in which functions are combined using a multiplexer with the goal to find an arrangement of the functions that yields the lowest cost. Specifically, we analyze two techniques to minimize total function cost 1) choosing which variables to apply to the multiplexer inputs and 2) choosing a permutation of logic values that yields lowest cost. We analyze the relative benefits of l), 2), and 1) and 2) together. Our basis of comparison is a set of randomly chosen two- and three-variable 4-valued functions. We show that these techniques yield a reduction of 7 to 34% in the average cost over the use of a multiplexer without such techniques.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**6**	

functions. We find that the application of techniques 1), 2), and 3) result a reduction of 7 to 34% in the average cost.

II. BACKGROUND AND NOTATION

Let $R = \{0, 1, \ldots, r-1\}$ be a set of r logic values, where $r \geq 2$, and let $X = \{x_1, x_2, , \ldots, x_n\}$ be a set of n variables, where x_i takes on values from R. A function $f(X)$ is a mapping $f: R^n \rightarrow R$. Let $U_{n,r}$ be the set of all n-variable r-valued functions. In the case of a one-variable function $f(x_1)$, it is convenient to use a vector notation, $f(x_1) = <f(0), f(1), \cdots, f(r-1)>$. For example, in one-variable 4-valued functions, <0,1,2,3> is the *identity* function and <3,2,1,0> is the *complement* function.

Let $c(f)$, the *cost* of function f, be a mapping $c: U_{n,r} \rightarrow \boldsymbol{R}$, where $\boldsymbol{R}$ is the set of real numbers. For example, the cost function $c(f)$ used in [5] correlates with the chip area occupied by the most compact implementation of f; it is called the *Area Cost*. We consider here two cost functions that correlate with the Area Cost, but are easily derived from the function specification [10]. They are as follows.

A. Transition Count - $TC(f)$

Given $f = <a_0, a_1, a_2, a_3>$, let

$$I_i(f) = \begin{cases} 1 \text{ if } a_{i-1} < a_i > a_{i+1} \text{ or } a_{i-1} > a_i < a_{i+1}, & 1 \leq i \leq 2 \\ 0 \text{ otherwise,} \end{cases}$$

$$I_{12}(f) = \begin{cases} 1 \text{ if } a_0 < a_1 = a_2 > a_3 \\ 0 \text{ otherwise, and} \end{cases}$$

$$ID(f) = \begin{cases} 1 \text{ if there is a } p, \ 0 \leq p \leq 2, \text{ such that } a_0 = a_1 = \cdots = a_p > a_{p+1}, \\ 0 \text{ otherwise.} \end{cases}$$

The *transition count* $TC(f)$ of function f is

$$TC(f) = I_1(f) + I_2(f) + I_{12}(f) + ID(f).$$

$TC(f)$ is the number of times the logic values in f change from decreasing to increasing and vice versa plus 1 if the function is initially decreasing. For example, $TC(<1122>) = 0$ and $TC(<2031>) = 3$.

B. Total Transition Size - $TTS(f)$

Given $f = <a_0, a_1, a_2, a_3>$, define beginning, middle, and end transition sizes as follows,

$$S_b(f) = \begin{cases} |a_1 - a_0| \text{ if } I_1 = 1 \text{ or } I_{12} = 1 \\ |a_2 - a_0| \text{ if } I_1 = 0 \text{ and } I_2 = 1 \\ |a_3 - a_0| \text{ otherwise,} \end{cases}$$

$$S_m(f) = \begin{cases} |a_2 - a_1| \text{ if } I_1 = I_2 = 1 \\ 0 \quad \text{otherwise, and} \end{cases}$$

$$S_e(f) = \begin{cases} |a_3 - a_2| \text{ if } I_2 = 1 \text{ or } I_{12} = 1 \\ |a_3 - a_1| \text{ if } I_2 = 0 \text{ and } I_1 = 1 \\ 0 \quad \text{otherwise.} \end{cases}$$

The *total transition size* $TTS(f)$ of a function f is

$$TTS(f) = S_b(f) + S_m(f) + S_e(f) + S_b(f)\, ID(f).$$

$TTS(f)$ is the sum of the size of each transition (increasing to decreasing or decreasing to increasing) plus the size of the first transition (again) if f is initially decreasing. For example, $TTS(<1122>) = 1$ and $TTS(<2031>) = 9$.

The analysis reported is based on the use of a specific costtable. The *universal costtable* is the set of all functions, $U_{n,r}$. For the case of one-valued 4-valued functions, $U_{1,4}$ consists of 256 functions. In the universal costtable, synthesis is simply a table lookup. In analyzing the effectiveness of 1) choosing which function variable to apply to the primary multiplexer inputs, of 2) permuting variable logic values, and of both 1) and 2), we use the universal costtable.

III. ANALYSIS OF MINIMIZATION TECHNIQUES

The realization of functions on two or more variables can be accomplished by multiplexers. Fig. 1 below shows the realization of a two-variable function as four one-variable functions and a multiplexer. The four functions

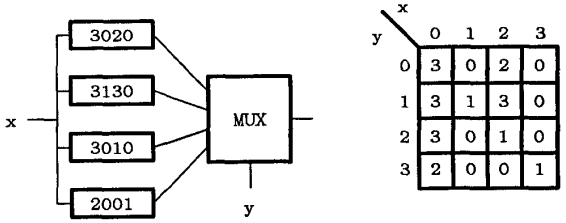

Figure 1. Realization of a function by multiplexers.

occur as rows in the map each depending on x. y is used to select which function is in effect. The total cost is the cost of the four functions plus the multiplexer cost. Using transition count, the four functions cost $3+3+3+2=14$.

The variable applied to the one-variable functions is the *primary* variable. In this example, it is x. If y is the primary variable, the columns represent the one-variable functions that must be realized, and x is used to select which function is in effect. For this case, the four functions cost $1+1+1+0=3$. Thus, there is a significant reduction when y is the primary variable.

Another way to reduce realization cost is by a *permuter* function to permute the variable value. Fig. 2 shows how the addition of a permuter function reduces the total cost. In this example, the total cost of the realization is the sum of the costs of the four functions, the multiplexer, and the permuter function. A reduction in total

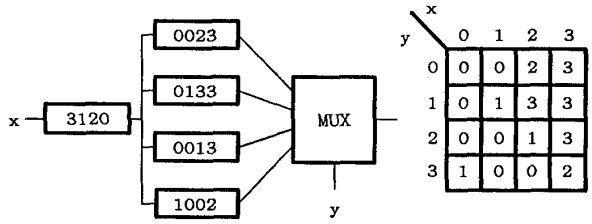

Figure 2. Using permuter functions to reduce the cost.

cost is realized only if the reduction in total cost of the four one-variable functions exceeds the cost of the additional permuter function. Here, there is a reduction; the cost of the four functions and the permuter function is $0+0+0+2+3=5$.

To analyze the relative benefit of choosing primary variables and permuting variable values, 5000 random functions were generated and their costs computed. We take as the cost the sum of the costs of the component one-variable functions plus the cost of the permuter function if used. The cost of the multiplexer is not included; it is assumed to be the same for all realizations. Depending on the technology used and the specific design, a savings in the multiplexer cost may be achieved with certain arrangements of functions. For example, if two inputs to the multiplexer represent the same function, there can be one less input and multiplexer cost can be lower. Similarly, in CCD, if not all multiplexer inputs must handle a logic 3, the multiplexer cost can be lower than for the case where all inputs must handle a logic 3. Since our goal is to analyze a logic design technique independent of the technology used (and thus the multiplexer design), we have not counted multiplexer cost. For each function f, the following costs are computed

1. The cost of f (the sum of the individual one-variable functions that compose f).
2. The least cost obtained from all ways to choose the primary variable of f.
3. The least cost obtained from all (24) ways to permute the primary variable values of f.
4. The least cost obtained from all ways to choose the primary variable of f and from all (24) ways to permute the value of the primary variable.

The analysis was performed on three sets of 5000 functions, one-, two-, and three-variable functions. See Figs. 3, 4, and 5. For one-variable functions, only the statistics of the costs in 1) were analyzed, since 2), 3), and 4) offer no advantage. This served to evaluate the random number generator used. The average cost and distribution of costs are known and so a comparison could be made. For functions on more variables, however, the average costs and distributions are *not* known, and the analysis provides previously unknown results. For two- and three-variable functions, 5000 functions is a small fraction of the total set. However, different sets of random functions produce nearly the same distributions, indicating a sufficiently large sample set size. Two cost functions are used 1) transition count and 2) total transition size. For the latter case, we take as the cost of the identity function, 0. That is, $TTS(<0,1,2,3>)=0$. This represents the view that the cost of $f(x)=x$ is not 3 as obtained by a direct calculation of total transition size, but 0, it being the direct application of the variable. The analysis accommodates the savings obtained in functions with duplicate one-variable functions. For example, if a two-variable function consists of four functions, two of which are the same, the duplicate function counts once, not twice. Here, the function output must drive more than one multiplexer input. Depending on the technology, a fanout unit may be required. We neglect the cost of this, because it is technology dependent. Figs. 3, 4, and 5 show the results for one-, two-, and three-variable functions. In all figures, the vertical axis shows the number of samples and the horizontal axis represents the cost. Fig. 3 shows the number of one-variable functions generated for each of the costs shown along the horizontal axis. The transition count is shown on the left side, and the total transition size is shown on the right side for the same set of 5000 random functions. Also shown in Fig. 3 (as dotted lines) is the distribution if the process was perfectly random, each function as likely as any other function. This was derived by calculating the number of functions with various values of the transition count and total transition size and prorating to a sample set of 5000. As can be seen, there is a close correlation with the distribution obtained from our sample set of 5000 functions. This is evidence that there

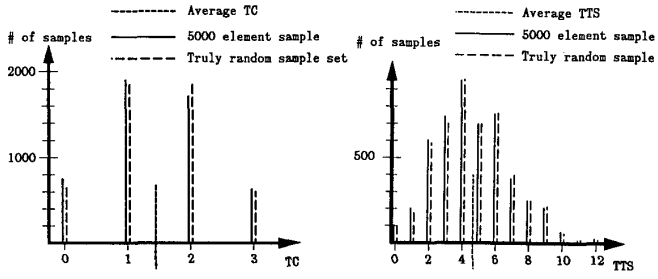

Figure 3. Statistics for one-variable 4-valued functions.

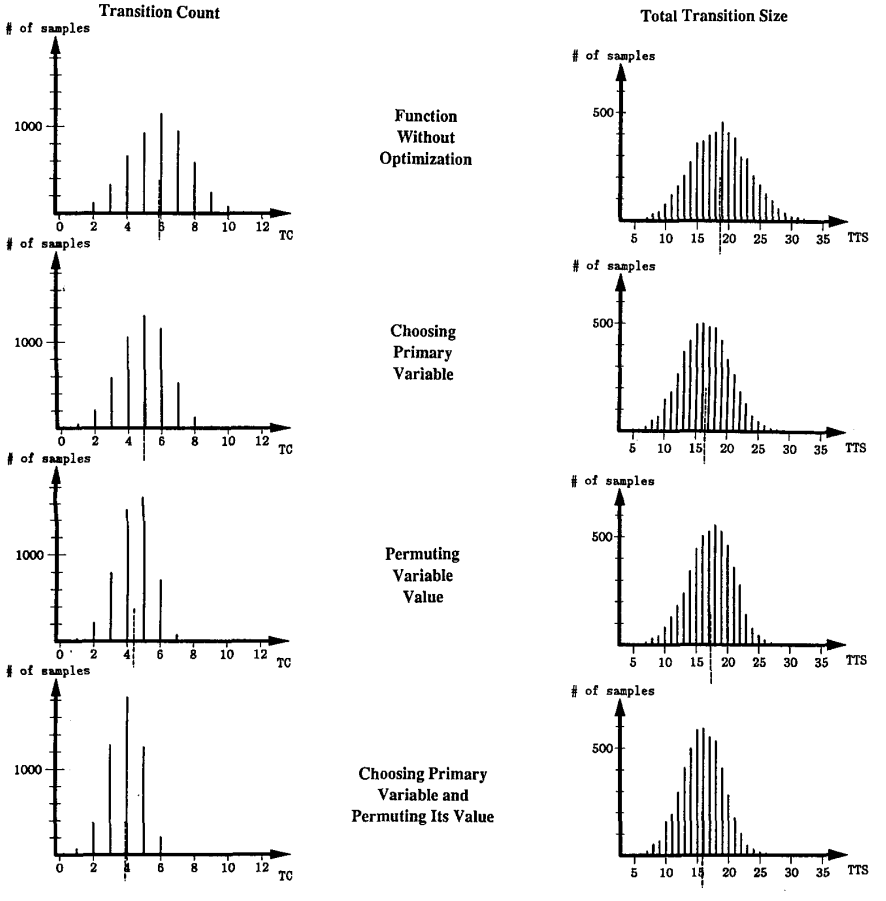

Figure 4. Statistics for two-variable 4-valued functions.

is little bias in the sample set.

A vertical dotted line extending above and below the horizontal axis in each figure shows the average cost. Figs. 4 and 5 shows the distribution of costs for 2- and 3-variable functions, respectively. The uppermost plots in the figure show the distribution of costs for randomly generated functions without optimization for the transition count (left side) and the total transition size (right side). The results of choosing a primary variable in all ways and picking the least cost realization is shown in the plots just below. For the transition count, there is about a 15% reduction in average cost in both the two- and three-variable cases (cf. Table I). For the total transition count, the reduction for both cases is about 10%. The third pair of plots from the top shows the result of permuting the primary variable values only. The reduction for the transition count is 25% and 18% for the case of two- and three-variable functions, respectively. For the total transi-

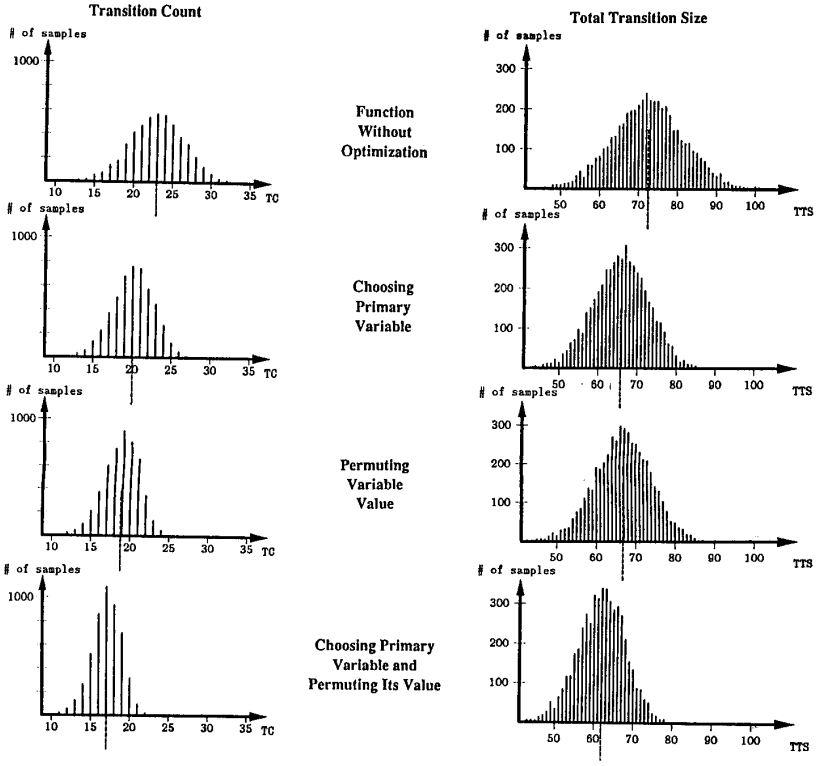

Figure 5. Statistics for three-variable functions.

Various Optimizations	Two-Variable Functions		Three-Variable Functions	
	Transition Count /% of *FWO*	Total Transition Size/% of *FWO*	Transition Count /% of *FWO*	Total Transition Size/% of *FWO*
Function Without Optimization (*FWO*)	5.88/100%	18.68/100%	22.95/100%	72.55/100%
Choosing Primary Variable	4.93/84%	16.49/88%	20.05/87%	65.77/91%
Permuting Variable Value	4.40/75%	17.30/93%	18.82/82%	66.74/92%
Choosing Primary Variable and Permuting its Value	3.89/66%	15.81/85%	17.02/74%	61.88/85%

Table I. Average Cost Over 5000 Functions

tion size, permuting the primary variable value yields a reduction of 7% and 8%, respectively.

The lowest plots show the distribution of average costs obtained by choosing the primary variable and permuting its value choosing the least cost realization. For the transition count, the reduction in average cost is 34% and 26% for two- and three-variable functions, respectively. For the total transition count, the reduction is 15% in both cases.

Table I shows the average costs of functions without optimization (*FWO*) together with the average costs obtained by optimizations 1), 2), and 3) discussed above. In addition, after each average cost is a percentage that represents the percentage the corresponding value to the average cost associated with *FWO*.

IV. COMPLEXITY OF THE SEARCH

The search for 1) a variable to apply to the select input of the multiplexer is straightforward. Given n variables, there are n ways to choose which to apply to the primary input. Having made this choice, there are then r^{n-1} one-variable functions whose cost must be looked up, for a total of $n\,r^{n-1}$ operations. Thus, for a fixed r, the time complexity of 1) is exponential in n.

The complexity of the search for 2) a permutation of logic values that produces the lowest cost realization can be characterized as follows. The number of permutations of the primary variable is $r!$, while the number of table look-up operations of costs of the resulting functions is r^{n-1}. There is another look-up operation for the cost of the permuter function, and so the time complexity for this case is $r!\,(r^{n-1}+1)$, which is also exponential in n.

The complexity of 1) selecting the primary variable and of 2) permuting its logic value is computed as follows. There are n ways to choose the variable and $r!\,(r^{n-1}+1)$ ways to permute the logic values and perform a table look-up of the costs of the component functions, for a total of $n\,r!\,(r^{n-1}+1)$ operations. The exponential time complexity of all three of these searches was seen clearly in our program; the time to analyze the case for $n=3$ was significantly larger than for $n=2$.

V. CONCLUDING REMARKS

Our initial expectation was that the permutation of variable values would result in larger improvements as the number of variables increased because the cost of the permuter function would be a smaller percentage of the total cost. However, this is not the case, as is seen by comparing the percentages for two- and to the percentages for three-variable functions, in Table I. It seems that the overriding factor is the increase in the number of one-variable functions, which offers fewer arrangements of advantageous logic value permutations. For example, when there is a large number of one-variable functions, any one permutation of variable values is as likely to yield the same cost as any other permutation.

VI. ACKNOWLEDGEMENTS

The authors thank the referees for useful comments.

REFERENCES

[1] M. H. Abd-El-Barr, Z. G. Vranesic, and S. G. Zaky, "Synthesis of MVL functions for CCD implementations," *Proceedings of the 16th International Symposium on Multiple-Valued Logic*, Blacksburg, VA, May 1986, pp. 116-127.

[2] M. H. Abd-El-Barr, T. D. Hoang, and Z. G. Vranesic, "The incremental cost approach for synthesis of CCD 4-valued unary functions," *Proceedings of the 18th International Symposium on Multiple-Valued Logic*, Palma de Mallorca, Spain, May 1988, pp. 82-89.

[3] J. T. Butler and H. G. Kerkhoff, "Analysis of input and output configurations for use in four-valued CCD programmable logic arrays," *IEE Proceedings*, **134**, Pt. E, 1987 No. 4, pp. 168-176.

[4] H. G. Kerkhoff, "Theory, design, and applications of digital charge-coupled devices," Ph.D. Thesis, Twente University of Technology, Enschede, The Netherlands, April 1984.

[5] H. G. Kerkhoff and H. A. J. Robroek, "The logic design of multiple-valued logic functions using charge-coupled devices," *Proceedings of the 12th International Symposium on Multiple-Valued Logic*, Paris, France, May 1982, pp. 35-44.

[6] J.-K. Lee and J. T. Butler, "Tabular methods for the design of CCD multiple-valued logic," *Proceedings of the 13th International Symposium on Multiple-Valued Logic*, Kyoto, Japan, May 1983, pp. 162-170.

[7] S. Onneweer, H. G. Kerkhoff, and J. T. Butler, "Structural computed-aided design of current-mode CMOS logic circuits," *Proceedings of the 18th International Symposium on Multiple-Valued Logic*, Palma de Mallorca, Spain, May 1988, pp. 21-30.

[8] K. A. Schueller and J. T. Butler, "The costtable realization is NP-complete," preprint.

[9] K. A. Schueller and J. T. Butler, "On the design of cost-tables for realizing multiple-valued circuits," preprint.

[10] K. A. Schueller, P. P. Tirumalai, and J. T. Butler, "An analysis of the costtable approach to the design of multiple-valued circuits," *Proceedings of the 16th International Symposium on Multiple-Valued Logic*, Blacksburg, VA, May 1986, pp. 42 - 50.